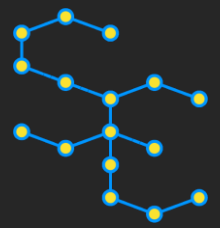

SEUPD@CLEF: Team Kueri on Argument Retrieval for Comparative Questions

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Marco Gallo Odai Mohammad Giacomo Virginio

Introduction

Task and Data

The task

Argument Retrieval for Comparative Questions

- Support users facing some choice problem from "everyday life“
- retrieve relevant argumentative passages for either compared object or for both

Collections

- Provided by CLEF
  - Topics-task2.xml which contains the topics.
  - DocT5Query expanded version of passages.jsonl which contains the documents expanded with queries generated using DocT5Query.
- Other collections
  - Historical stoplists: lucene, smart and terrier;
  - Custom stoplists:
    - Kueristop - Stoplist formed by the 400 most concurrent term in the Contents field of the document collection;
    - Kueristopv2 - Subset of kueristop, obtained by removing from it terms appearing in the Objects field of the topics, except for the very general terms also appearing in lucene stoplist ("in" and "the").
  - Sentence quality

Manual relevance

- The conventional and ideal approach when evaluating the performance of the runs would have been to use last year's test collection.
- Since we did not have access to last year's corpus we have decided to use this year's test collection to evaluate our systems, using a *qrels* file containing relevance feedback manually performed by us.
- The *qrels* file has been built by gathering, for each of the runs performed, the top 5 ranked documents for each topic.

Approach

The system developed

Approach

- We developed a system based on Apache Lucene.
- The system is made up of the following components:
  - Parse
  - Analyze
  - Index
  - Search
  - Relevance Feedback
  - Reciprocal Ranking Fusion
  - Argument quality

Parse

- This stage is divided into two parts:
  - Parsing documents, each document has three attributes:
    - Document ID
    - Document content
    - Document DocT5Query
  - Parsing Topics, each topic has three attributes:
    - Number
    - Title
    - Objects

Analyze

- We have built multiple analyzers
  - To process the documents
  - Perform tokenization
  - Use different combinations of filters
  - Use different combinations of stoplists

Index

- We performed indexing with different combinations of
  - Tokenizers
  - Similarities
- For each document we indexed:
  - The document ID
  - The document content
  - The DocT5Query field

Search

- Retrieving and preparing the topics for the search.
- Defining how to use topics in the search.
  - We used topics titles in the search by similarity
  - then we added the topic objects as a MUST clause in the search to act as “filters”
- Defining which type of comparison to perform between topics and documents.
- Writing the results on a file

Relevance Feedback (RF)

- RF is a technique to perform query expansion
- The tokens and their frequency in the relevant documents are retrieved by searching the document by docID and iterating through its termvector.
- The tokens used in the search are boosted by their frequency in the document multiplied by the square of the relevance score.
- We used the following customized formula for the Rocchio Algorithm

$$\overrightarrow{Q_m} = k_i \cdot \frac{1}{|D_r|} \cdot \sum_{\overrightarrow{D_i} \in D_r} \overrightarrow{D_i}$$

Reciprocal Ranking Fusion (RRF)

- Perform RRF using all the runs in .txt documents inside the input directory.
- For each document and for each topic, the documents and their respective ranking are collected.
- Then each document receives a new scoring using the RRF formula

$$RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)}$$

Argument Quality

- We decided to make use of IBM Project Debater API, especially the argument quality service of the API.
- We used the API, for each passage in the corpus, to gage the quality of the sentence as a text, which means how good they are written.
- Then used the obtained scores to rerank the results of the search saved in a run file.

Results

System performance

Results

Table 1
NDCG@5 and setup for single runs

#	NDCG@5	num_q	RF	Stoplist	Filter	Stemmer	Similarity	Weights	Reranking
1	0.3830	50	False	lucene	False	None	BM25	[1,1]	False
2	0.3756	50	False	lucene	False	None	LMD	[1,1]	False
3	0.3313	50	False	lucene	False	None	TFIDF	[1,1]	False
4	0.4140	50	False	smart	False	None	BM25	[1,1]	False
5	0.4258	50	False	terrier	False	None	BM25	[1,1]	False
6	0.4366	50	False	kueristop	False	None	BM25	[1,1]	False
7	0.4548	50	False	kueristopv2	False	None	BM25	[1,1]	False
8	0.4015	48	False	lucene	True	None	BM25	[1,1]	False
9	0.4759	41	False	kueristop	True	None	BM25	[1,1]	False
10	0.4823	48	False	kueristopv2	True	None	BM25	[1,1]	False
11	0.2634	50	False	kueristopv2	False	None	BM25	[0,1]	False
12	0.3654	50	False	kueristopv2	False	None	BM25	[1,0]	False
13	0.4525	50	False	kueristopv2	False	None	BM25	[1,2]	False
14	0.4674	50	False	kueristopv2	False	None	BM25	[2,1]	False
15	0.4873	50	False	kueristopv2	False	Porter	BM25	[1,1]	False
16	0.8549	50	True	kueristopv2	False	False	BM25	[1,1]	False
17	0.8552	50	True	kueristopv2	False	Porter	BM25	[1,1]	False
18	0.5867	48	False	kueristopv2	True	None	BM25	[1,1]	True
19	0.5392	50	False	kueristopv2	False	None	BM25	[2,1]	True
20	0.5714	50	False	kueristopv2	False	Porter	BM25	[1,1]	True
21	0.8606	50	True	kueristopv2	False	False	BM25	[1,1]	True
22	0.8323	50	True	kueristopv2	False	Porter	BM25	[1,1]	True

Table 2
NDCG@5 and setup for rrf runs

# fused	NDCG@5	Reranking
10,14,15,16,17	0.7521	False
10,14,15,16,17	0.7450	True

Results

- Compare similarity functions
- Compare stop lists
- Compare impact of filtering
- Changing the weight of Contents and DocT5Query fields respectively
- Experimenting with adding a stemmer vs not adding one
- Relevance Feedback
- Reranking with argument quality

Failure analysis

- In the first iteration of manual scoring, no relevant documents were found for topic77 “Is it healthier to bake than to fry food?”
- Lots of documents retrieved were just ads and were not relevant, but argument quality solves this problem.

Conclusion

And future work

Conclusion

- Tested out different techniques and tools studied in lessons, to discover new ones on our own and experiment with their impact in a "real world" application.
- Having access to last year's corpus would allow us for example to
  - Fine tune BM25 parameters
  - Fine tune the field weights
  - Fine tune the boosts for terms in RF
  - And we could experiment with many more stoplists and stemmers.
- In future works it would be interesting to experiment with machine learning and deep learning techniques, that have become the standard in the last decade of information retrieval

Thank you